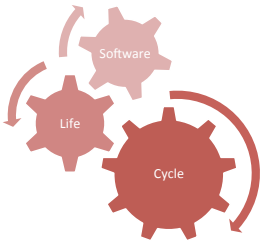
Software Development Lifecycle

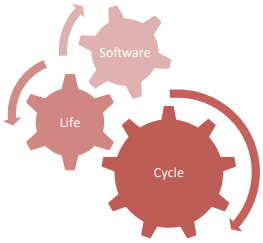


thinking about the process

Next week

- Monday:
  - interactive section on extreme programming
  - Play SimSE (think SimCity but with software engineers)
  - Bring a laptop!

Software Development Lifecycle



thinking about the process

How complex is software?

How complex is software?

- Measures of complexity:
  - lines of code
  - number of classes
  - number of modules
  - module interconnections and dependencies
  - time to understand
  - # of authors
  - … many more

Windows Server 2003: 50 MSLoC
Debian 5.0: 324 MSLoC

- Google keeps all their code in a single repository, all at HEAD
- Sept 16, 2015 WIRED article reported that code is 2 billion lines of code

http://www.wired.com/2015/09/google-2-billion-lines-codeand-one-place/

CADE METZ BUSINESS 09.16.15 10:00 AM

GOOGLE IS 2 BILLION LINES OF CODE—AND IT'S ALL IN ONE PLACE

GETTY IMAGES

HOW BIG IS Google? We can answer that question in terms of revenue or stock price or customers or, well, metaphysical influence. But that's not all. Google is, among other things, a vast empire of computer software. We can answer in terms of code.

Google's Rachel Potvin came pretty close to an answer Monday at an engineering conference in Silicon Valley. She estimates that the software needed to run all of Google's Internet services—from Google Search to Gmail to Google Maps—spans some 2 billion lines of code. By comparison, Microsoft's Windows operating system—one of the most complex software tools ever built for a single computer, a project under development since the 1980s—is likely in the realm of 50 million lines.

So, building Google is roughly the equivalent of building the Windows operating system 40 times over.

Managing software development

- Requirements
- Design
- Implementation
- Testing
- Maintenance

Outline

- Why do we need a lifecycle process?
- Lifecycle models and their tradeoffs
  - code-and-fix
  - waterfall
  - spiral
  - staged delivery
  - agile (scrum)
  - … there are many others

Ad-hoc development

- Creating software without any formal guidelines or process
- Advantage: easy to learn and use!
- Disadvantages?

Ad-hoc development disadvantages

- Some important actions (testing, design) may go ignored
- Unclear when to start or stop each task
- Scales poorly to multiple people
- Hard to review or evaluate one's work

The later a problem is found in software, the more costly it is to fix.

What makes a lifecycle?

- Requirements
- Design
- Implementation
- Testing
- Maintenance

How do we combine them?

Benefits of using a lifecycle

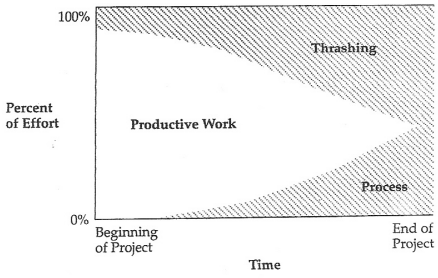
- provides a work structure
- forces thinking about the "big picture"
- helps prevent decisions that are individually on target but collectively misdirected
- assists management and progress control

What are some drawbacks?

Are there analogies outside of SE?

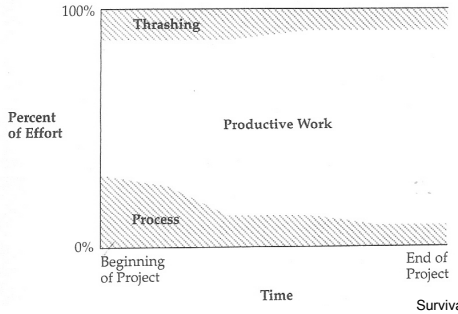
Consider the process of building the Prudential

Project with little attention to process



Survival Guide: McConnell p24

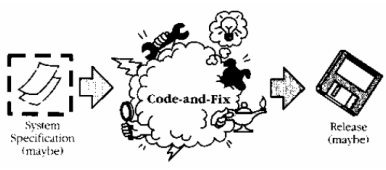
Project with early attention to process



Survival Guide: McConnell p25

Let's talk about some lifecycle models

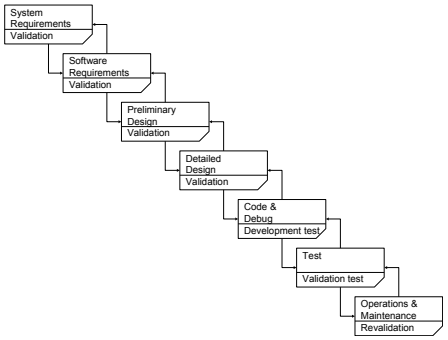
Code-and-fix model



Code-and-fix model

- Advantages
  - Low overhead
  - Applicable to small, short-lived projects
- Dangers
  - No way to assess progress and manage risks
  - Hard to accommodate changes
  - Unclear what and when will be delivered
  - Hard to assess quality

Waterfall model



Waterfall model advantages

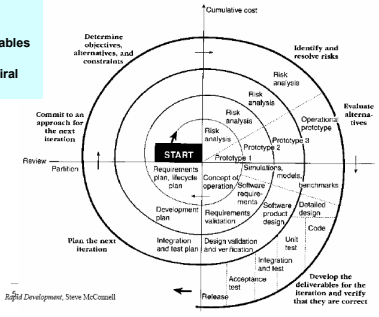
- Works well for well-understood projects
  - tackles all planning upfront
  - no midstream changes leads to efficient software development process
- Supports experienced teams
  - Orderly, easy-to-follow sequential model
  - Reviews help determine readiness to advance

Waterfall model limitations

- Difficult to do all planning upfront
- No sense of progress until the end
- Integration occurs at the very end
  - Defies the "integrate early and often" rule
  - Without feedback, solutions are inflexible
  - Final product may not match customer's needs
- Phase reviews are massive affairs
  - It takes a lot of inertia and $ to make changes

Spiral model

Determine objectives
Identify and resolve risks
Evaluate alternatives
Develop and verify deliverables
Plan next spiral
Commit (or not) to next spiral



Rapid Development, Steve McConnell

Spiral model

- Oriented towards phased reduction of risk
- Take on the big risks early
  - are we building the right product?
  - do we have customers for this product?
  - is it possible to use existing technology?
    - tomorrow's technology?
- Progresses carefully toward a result

Spiral model advantages

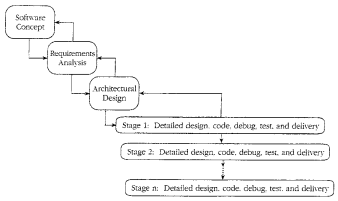
- Especially appropriate at the beginning of the project, allowing requirement fluidity
- Provides early indication of unforeseen problems
- Allows for change
- As costs increase, risks decrease!

Addresses the biggest risk first

Spiral model disadvantages

- A lot of planning and management
- Requires customer and contract flexibility
- Developers must be able to assess risk

Staged delivery model



first, waterfall-like
then, short release cycles: plan, design, execute, test, release
with delivery possible at the end of any cycle

Staged delivery model advantages

- Can ship at the end of any release cycle
- Intermediate deliveries show progress, satisfy customers, and lead to feedback
- Problems are visible early (e.g., integration)
- Facilitates shorter, more predictable release cycles

Very practical, widely used and successful

Staged delivery model disadvantages

- Requires tight coordination with documentation, management, marketing
- Product must be decomposable
- Extra releases cause overhead

What's the best model?

Consider

- The task at hand
- Risk management
- Quality / cost control
- Predictability
- Visibility of progress
- Customer involvement and feedback

Aim for good, fast, and cheap.
But you can't have all three at the same time.

Next week

- Monday:
  - interactive section on extreme programming
  - Play SimSE (think SimCity
    but with software engineers)
  - Bring a laptop!